
| RESEARCH ARTICLE

Data Lake Architecture at Uber: A Lambda-Based Approach to Real-Time and Batch Analytics with Cross-Industry Perspectives

Piyush Dubey

University of Iowa, USA

Corresponding Author: Piyush Dubey, **E-mail:** piyushdubey.email@gmail.com

| ABSTRACT

The evolution of data infrastructure in modern transportation platforms demonstrates the critical role of Lambda architecture in addressing the dual challenges of real-time processing and comprehensive historical analytics. Through the implementation of sophisticated data lake architectures leveraging open-source technologies, including Apache Kafka for streaming, Apache Flink for real-time processing, Apache Hudi for data lake management, and Presto for distributed querying, organizations achieve significant reductions in data freshness latency while maintaining scalability. The architectural framework encompasses three fundamental layers: batch processing for accuracy and completeness, speed processing for low-latency insights, and a serving layer for unified query interfaces. Performance optimizations through smart query routing, multi-region deployments, and hierarchical caching enable sub-second response times for critical business decisions. Comparative examination across government, healthcare, retail, and automotive sectors reveals both convergent patterns in lakehouse adoption and sector-specific adaptations driven by regulatory requirements and operational constraints. Government implementations prioritize security and audit capabilities within hybrid cloud deployments, healthcare organizations emphasize privacy-preserving analytics for inventory optimization, while automotive manufacturers leverage edge-to-cloud architectures for vehicle telemetry processing. The synthesis of cross-industry implementations highlights essential success factors, including business-objective alignment, comprehensive data governance from inception, incremental migration strategies, and cultural transformation initiatives that complement technical deployments.

| KEYWORDS

Lambda architecture, data lake management, real-time stream processing, multi-region deployment, lakehouse architecture.

| ARTICLE INFORMATION

ACCEPTED: 12 June 2025

PUBLISHED: 06 July 2025

DOI: 10.32996/jcsts.2025.7.7.35

1. Introduction

The modern ride-sharing ecosystem operates across a vast global network spanning thousands of cities, generating enormous volumes of data from diverse sources, including location services, transaction systems, user interactions, and operational metrics. This massive scale presents unprecedented data infrastructure challenges, requiring platforms capable of processing petabytes of information daily while maintaining sub-second latency for critical business decisions. The complexity intensifies when considering the need for both immediate insights to support dynamic pricing algorithms and comprehensive historical analysis for strategic planning. These dual requirements have driven the evolution of sophisticated data architectures that can seamlessly blend real-time processing with batch analytics capabilities [1].

The Lambda architecture has emerged as a foundational pattern for addressing these challenges, providing a robust framework that maintains separate batch and speed processing layers to serve accurate, low-latency views over massive datasets. This architectural approach gains particular relevance in contemporary data analytics environments where organizations must handle the velocity, volume, and variety of big data while ensuring fault tolerance and horizontal scalability. The architecture's strength lies in its ability to process parallel pipelines, enabling organizations to deliver immediate insights for operational decisions while

Copyright: © 2025 the Author(s). This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) 4.0 license (<https://creativecommons.org/licenses/by/4.0/>). Published by Al-Kindi Centre for Research and Development, London, United Kingdom.

maintaining data accuracy and completeness for strategic analysis. Recent implementations have demonstrated how this pattern can be enhanced through modern stream processing technologies and unified data lake management systems that bridge the gap between batch and streaming workloads [2].

The evolution of data lake architectures in the transportation sector exemplifies how organizations can effectively balance real-time and batch processing needs through the strategic implementation of open-source technologies. By leveraging stream processing frameworks for data ingestion, real-time analytics engines for immediate insights, timeline-consistent storage layers for efficient data lake management, and distributed query engines for analytics, modern platforms achieve significant reductions in data freshness latency. This architectural approach supports multiple analytics categories, including streaming analytics for immediate updates, real-time analytics for customer-facing features, interactive analytics for operational reporting, and batch analytics for historical analysis. The implementation of such systems requires careful consideration of performance optimizations, including smart query routing, multi-region deployments, and sophisticated caching mechanisms [1].

The significance of these architectural patterns extends beyond the transportation industry, as evidenced by implementations across diverse sectors. Government agencies have deployed similar architectures to enhance operational efficiency and fraud detection capabilities, processing millions of transactions to identify anomalies and optimize service delivery. In the healthcare retail sector, data lake technologies enable the integration of disparate data sources, supporting real-time inventory management and personalized customer experiences across distributed locations. Meanwhile, the automotive industry leverages these architectures to process vast amounts of IoT data from connected vehicles, enabling predictive maintenance and enhanced driving experiences through real-time analytics. These varied implementations underscore the versatility and critical importance of modern data lake architectures in addressing sector-specific requirements while maintaining common architectural principles [2].

2. Theoretical Framework and Architecture Design

The Lambda architecture represents a comprehensive data processing paradigm that addresses the fundamental challenge of serving accurate, low-latency views over massive datasets through a three-layer approach. The batch layer functions as the immutable system of record, processing append-only raw data to generate comprehensive batch views that prioritize accuracy and completeness. This layer leverages distributed computing frameworks to perform exhaustive computations across entire datasets, though inherent processing delays exist due to the time required for complete data traversal and computation. The speed layer addresses these latency constraints by processing only the most recent data streams in real-time, generating approximate views that trade some degree of accuracy for immediate availability. These real-time computations remain temporary, serving queries until the more accurate batch computations complete and supersede them. The serving layer synthesizes outputs from both processing paths, indexing and exposing merged results through a unified query interface that abstracts the underlying architectural complexity. This design pattern ensures fault tolerance through recomputation capabilities while maintaining eventual consistency, as the immutable nature of the batch layer enables complete system recovery and historical reprocessing when necessary [3].

Modern implementations of this architectural pattern depend on sophisticated open-source technologies that collectively enable seamless integration between batch and streaming paradigms. The data ingestion layer typically employs distributed messaging systems that function as persistent, replayable logs, enabling multiple processing pipelines to consume identical data streams independently. These systems support both real-time event processing and reliable data replay for batch operations through their publish-subscribe architectures. Stream processing engines provide the computational backbone for the speed layer, offering advanced capabilities including exactly-once processing semantics, stateful computations, and sophisticated windowing operations that enable complex event processing while maintaining consistency guarantees. The data lake management layer introduces critical capabilities for handling large-scale analytical datasets on distributed storage systems, including atomic write operations, snapshot isolation for consistent reads, and incremental processing capabilities that blur traditional boundaries between batch and streaming workloads. Distributed SQL engines complete the technology stack by providing familiar query interfaces that abstract storage heterogeneity, enabling interactive analytics across diverse data formats and locations without requiring users to understand underlying implementation details [4].

Large-scale data platforms require sophisticated workload categorization to optimize resource allocation and meet diverse analytical requirements effectively. The streaming analytics tier represents the most latency-critical workload category, processing continuous data streams to enable real-time decision-making capabilities such as dynamic pricing adjustments and anomaly detection systems. These workloads demand sub-second processing latencies and require specialized infrastructure optimizations to maintain consistent performance under varying load conditions. Real-time analytics occupies a middle ground, operating with slightly relaxed latency requirements while still maintaining responsiveness for customer-facing features and operational dashboards. This tier typically processes data within seconds to minutes, balancing computational efficiency with

timeliness to support features like recommendation systems and live operational metrics. Interactive analytics serves analytical users who require rapid exploration capabilities over large datasets, expecting query responses within seconds to support iterative analysis and dashboard generation. These workloads benefit from sophisticated query optimization and caching strategies to maintain responsiveness across diverse query patterns. Batch analytics handles comprehensive historical processing, machine learning model training, and complex aggregations where accuracy and completeness supersede latency considerations, often running for extended periods to process massive datasets thoroughly [3].

The architectural framework extends beyond pure technical implementation to encompass critical design considerations that ensure long-term sustainability and operational efficiency. Schema evolution capabilities prove essential in dynamic environments where data producers and consumers evolve independently, requiring sophisticated versioning and compatibility mechanisms to prevent breaking changes from disrupting production systems. Data lineage tracking becomes paramount for maintaining data quality and supporting regulatory compliance, necessitating comprehensive metadata management systems that capture transformation logic and processing history throughout the pipeline. Multi-tenancy support influences architectural decisions across all layers, from resource isolation strategies in stream processing frameworks to fair scheduling algorithms in query engines, ensuring equitable resource distribution while maintaining performance isolation between different workload classes. These considerations collectively shape implementation decisions and operational practices, determining the ultimate success of large-scale data platforms in meeting diverse analytical needs while maintaining operational efficiency and system reliability [4].

| Layer | Primary Function | Processing Type | Latency | Accuracy | Data Retention |
|---------------|--|--|-------------------------------|------------------|----------------------------------|
| Batch Layer | System of record, comprehensive processing | Batch processing on the complete dataset | High (hours to days) | Maximum accuracy | Permanent, immutable |
| Speed Layer | Real-time insights, recent data only | Stream processing | Low (milliseconds to seconds) | Approximate | Temporary until batch catches up |
| Serving Layer | Query interface, merged views | Indexing and serving | Low (milliseconds) | Best available | As configured |

Table 1: Lambda Architecture Layer Characteristics [3]

3. Implementation and Technical Optimizations

Real-time processing pipelines form the backbone of modern data architectures that require immediate responses to rapidly changing conditions in dynamic marketplaces. These sophisticated systems continuously ingest streams of location updates, transactional events, and user interaction data through distributed messaging platforms, applying complex event processing algorithms to derive actionable insights within milliseconds of data generation. The architecture leverages stateful stream processing capabilities to maintain rolling aggregations and temporal windows, enabling critical features such as dynamic pricing mechanisms that must continuously evaluate supply-demand equilibrium across multiple geographic zones. Implementation of exactly-once processing semantics ensures data integrity while maintaining ultra-low latency, utilizing advanced checkpointing mechanisms and distributed state management to recover seamlessly from failures without introducing data loss or duplication. The system employs various windowing strategies, including tumbling windows for discrete time intervals, sliding windows for overlapping computations, and session windows for user-activity-based grouping, each serving specific analytical needs. These windowing operations enable sophisticated time-based aggregations that power features ranging from demand prediction to automated resource allocation decisions. The pipeline architecture incorporates adaptive backpressure handling and elastic scaling capabilities, automatically adjusting computational resources in response to fluctuating data volumes and processing complexity, ensuring consistent performance even during peak load periods [5].

| Characteristic | Traditional Data Warehouse | Modern Data Lake | Lakehouse Architecture |
|------------------|----------------------------|---|--|
| Data Types | Structured only | All types (structured, semi-structured, unstructured) | All types with warehouse performance |
| Schema Approach | Schema-on-write | Schema-on-read | Hybrid approach |
| Processing Model | Batch-centric | Batch and stream | Unified batch and stream |
| Cost Model | High compute and storage | Cost-effective storage | Optimized computing with cheap storage |
| Use Case Focus | Business intelligence | Exploratory analytics, ML | All analytical workloads |
| Data Freshness | Hours to days | Near real-time possible | Real-time to batch |

Table 2: Data Lake vs. Traditional Architecture Comparison [5, 8]

Batch processing workflows provide essential complementary capabilities for comprehensive historical analysis that would be computationally infeasible within streaming constraints. These systems operate on complete datasets residing in distributed storage systems, harnessing massive parallelism to process extensive historical records for pattern identification, trend analysis, and machine learning model development. The implementation orchestrates complex multi-stage processing through directed acyclic graph frameworks, ensuring proper dependency resolution and robust failure recovery across distributed computing clusters. Advanced partitioning strategies optimize data locality and minimize network transfer overhead, enabling parallel processing across numerous compute nodes while maintaining consistency through sophisticated coordination protocols. The architecture incorporates incremental processing paradigms to eliminate redundant computation on unchanged data segments, dramatically reducing both computational costs and processing duration for recurring analytical workloads. These batch systems generate comprehensive derived datasets that populate enterprise data warehouses and analytical platforms, forming the foundation for business intelligence initiatives, compliance reporting, and long-term strategic planning. The integration between batch and streaming layers ensures that historical context enriches real-time decision-making while real-time insights inform batch processing priorities [6].

Performance optimization within large-scale data platforms demands comprehensive strategies addressing query execution efficiency, resource utilization, and geographic distribution challenges. Intelligent query routing mechanisms analyze incoming analytical requests to determine optimal execution strategies, evaluating factors including data freshness requirements, computational complexity, and resource availability. The routing layer maintains detailed statistics regarding data distribution patterns and historical query performance, enabling sophisticated decisions about serving results from real-time materialized views, pre-computed batch aggregates, or hybrid approaches that seamlessly blend both sources. Multi-region deployment architectures address latency minimization and availability requirements through strategic data replication and processing capability distribution across geographic locations, implementing nuanced consistency models that balance data freshness against network efficiency. The system employs eventual consistency for analytical workloads while maintaining strong consistency guarantees for critical operational data, utilizing advanced conflict resolution algorithms to handle concurrent modifications across regions. Caching implementations span multiple hierarchical levels, from distributed in-memory caches storing frequently accessed aggregations to persistent result caches for common analytical queries, substantially reducing both query latency and computational overhead. The caching strategy incorporates intelligent invalidation mechanisms balancing cache effectiveness with data freshness, employing both time-based expiration and event-driven invalidation appropriate to different data categories and usage patterns [5].

Modern data lake management technologies have revolutionized the traditional trade-offs between data completeness and analytical timeliness through innovative storage and processing paradigms. Timeline-consistent storage layers maintain comprehensive change histories for all data modifications, enabling efficient incremental processing by computing precise deltas between temporal snapshots. This approach dramatically reduces the latency gap between data generation and analytical availability, enabling near-real-time analytics on datasets traditionally confined to batch processing cycles. The implementation leverages copy-on-write semantics and merge-on-read strategies to optimize write throughput while preserving query performance, automatically compacting data files to prevent fragmentation and maintain optimal read characteristics.

Sophisticated indexing techniques, including bloom filters for existence checks and column-level statistics for predicate pushdown, enable efficient data skipping during query execution, minimizing data scanning requirements and accelerating query performance. The architecture supports concurrent read and write operations without blocking, facilitating continuous data ingestion while simultaneously serving analytical queries, fundamentally transforming organizational approaches to data freshness. These optimizations collectively achieve sub-minute data freshness for analytical workloads previously requiring extended batch processing windows, enabling organizations to make data-driven decisions with unprecedented timeliness and accuracy [6].

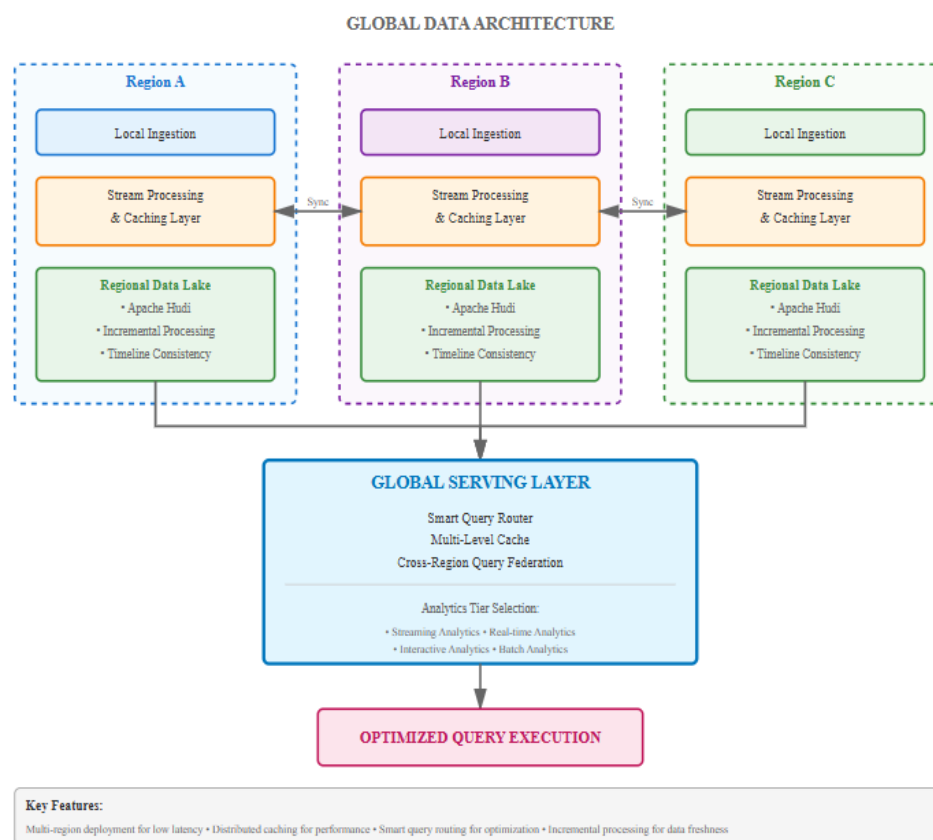


Fig. 1: Multi-Region Architecture with Performance Optimization Components [5, 6]

4. Comparative Industry Analysis

Government sector implementations of data lake architectures demonstrate sophisticated approaches to fraud detection and operational efficiency optimization within complex regulatory frameworks. Federal oversight agencies have deployed comprehensive data platforms that process diverse data streams, including transactional records, operational metrics, geospatial information, and behavioral patterns, to identify anomalies indicative of fraudulent activities or systemic inefficiencies. These architectures leverage distributed computing frameworks to analyze historical patterns while simultaneously processing real-time data feeds, enabling proactive intervention strategies that prevent issues from escalating into significant operational or financial impacts. The implementation incorporates multiple layers of security controls, including encryption at rest and in transit, role-based access controls, and comprehensive audit logging to meet stringent federal compliance requirements. Machine learning models operating within these environments must balance transparency requirements for government decision-making with the sophistication needed to detect evolving fraud patterns. The architecture's success in reducing investigation times while improving detection accuracy demonstrates how public sector organizations can modernize their analytical capabilities while maintaining necessary oversight and accountability standards. These implementations often utilize hybrid cloud deployments that keep sensitive data within government-controlled infrastructure while leveraging cloud elasticity for computational workloads [7, 9].

Healthcare retail analytics platforms present unique architectural challenges due to the intersection of commercial optimization needs with strict healthcare privacy regulations. Modern pharmaceutical retail chains have implemented data lake solutions that integrate point-of-sale systems, prescription management databases, supply chain platforms, and customer loyalty programs to create comprehensive analytical environments. These architectures must handle the complexity of inventory management across

distributed locations while ensuring compliance with healthcare regulations through sophisticated data governance frameworks. The systems process streaming transaction data alongside batch analytical workloads to enable real-time inventory optimization, demand forecasting, and personalized customer engagement strategies. Privacy-preserving techniques, including tokenization, differential privacy, and homomorphic encryption, enable analytics on sensitive health data without compromising individual privacy. Advanced analytics capabilities support seasonal demand prediction, regional health trend identification, and optimization of pharmaceutical stock levels based on demographic patterns and historical consumption data. The architecture's ability to reduce medication stockouts while minimizing inventory waste demonstrates the transformative potential of data lakes in highly regulated retail environments where customer health outcomes depend on product availability [8, 10].

The automotive industry's transition toward connected and autonomous vehicles has necessitated revolutionary approaches to data architecture, particularly in managing the massive volumes of telemetry data generated by modern vehicle fleets. Contemporary electric vehicle manufacturers have implemented sophisticated data lake infrastructures capable of ingesting and processing continuous streams of sensor data encompassing battery performance metrics, driving behavior patterns, environmental conditions, and component health indicators from thousands of vehicles operating simultaneously across diverse geographic regions. These architectures employ edge computing paradigms to perform initial data processing within vehicles themselves, reducing bandwidth requirements and enabling real-time safety features while centralizing complex analytics in cloud-based environments. The hybrid edge-cloud approach supports predictive maintenance algorithms that identify potential component failures through pattern recognition across fleet-wide data, enabling proactive service scheduling that maximizes vehicle availability. The infrastructure also facilitates over-the-air update mechanisms based on aggregated fleet insights, allowing continuous enhancement of vehicle performance, energy efficiency, and user experience features. This implementation paradigm demonstrates how traditional manufacturing companies can evolve into data-driven service providers, creating recurring revenue streams through subscription-based features and value-added services enabled by comprehensive data analytics [7, 11].

Analysis of implementations across diverse industries reveals both convergent patterns and sector-specific adaptations in data lake architecture design. Common architectural patterns include the adoption of lakehouse paradigms that combine data lake flexibility with data warehouse performance characteristics, enabling organizations to support diverse analytical workloads within unified platforms. The widespread implementation of open table formats reflects industry recognition of the importance of avoiding vendor lock-in while maintaining performance and functionality. Organizations across sectors are implementing comprehensive data catalogs with automated metadata management, enabling self-service analytics while maintaining governance standards. However, significant divergences emerge in implementation details driven by sector-specific constraints. Government implementations prioritize sovereignty and audit requirements, often necessitating specialized deployment models that maintain data within jurisdictional boundaries. Healthcare organizations emphasize fine-grained access controls and privacy-preserving computation techniques to protect sensitive patient information while enabling valuable analytics. Automotive implementations focus on edge-to-cloud architectures optimized for high-velocity sensor data processing with ultra-low latency requirements for safety-critical applications. These variations illustrate how organizational context and regulatory environments fundamentally shape architectural decisions beyond purely technical considerations [8].

| Industry Sector | Primary Use Cases | Key Technologies | Unique Requirements | Architectural Focus |
|-------------------|---|--|--|--|
| Government | Fraud detection, Operational efficiency | Hybrid cloud, ML frameworks | Federal compliance, Audit trails, Data sovereignty | Security-first, Traceable decisions |
| Healthcare Retail | Inventory optimization, Customer insights | Privacy-preserving analytics, Real-time processing | HIPAA compliance, Patient privacy | Privacy by design, Distributed inventory |
| Automotive | Vehicle telemetry, Predictive maintenance | Edge computing, IoT platforms | Ultra-low latency, Safety-critical processing | Edge-to-cloud hybrid, Real-time response |
| Transportation | Dynamic pricing, Route optimization | Stream processing, Real-time analytics | Sub-second latency, Scale elasticity | Performance optimization, Multi-region |

Table 3: Industry-Specific Data Lake Implementation Characteristics [7, 8]

The synthesis of experiences across these diverse implementations yields critical insights for organizations that are planning data lake initiatives. Successful deployments consistently demonstrate the importance of establishing clear business objectives that drive architectural decisions rather than pursuing technology for its own sake. Organizations achieving sustainable value from data lake investments prioritize comprehensive data governance frameworks from inception, including data quality monitoring, lineage tracking, and lifecycle management capabilities. The democratization of data access through intuitive self-service platforms emerges as a critical success factor, requiring careful balance between accessibility and security. Successful implementations adopt incremental migration strategies that allow organizations to learn and adapt rather than attempting comprehensive transformations that risk disruption. The importance of cultural transformation alongside technical implementation becomes evident, with leading organizations investing significantly in data literacy programs and change management initiatives. Perhaps most critically, while architectural patterns provide valuable starting points, successful implementation requires thoughtful adaptation to sector-specific requirements, whether addressing regulatory compliance in healthcare, security requirements in government, or performance demands in automotive applications [7].

5. Conclusion

The transformation of data architecture in the transportation sector through Lambda-based implementations represents a paradigm shift in how organizations balance real-time operational needs with comprehensive analytical requirements. The successful integration of streaming and batch processing layers, supported by modern open-source technologies, enables platforms to achieve sub-minute data freshness for workloads previously confined to extended batch processing windows. The architectural patterns demonstrated transcend industry boundaries, with government, healthcare, and automotive sectors adapting core principles to meet sector-specific requirements while maintaining common foundations. The evolution from traditional data warehouses to modern lakehouse architectures reflects broader technological trends toward unified platforms that eliminate historical trade-offs between flexibility and performance. Organizations embarking on similar transformations must recognize that technological implementation alone cannot guarantee success; rather, the combination of clear business objectives, robust governance frameworks, incremental migration strategies, and organizational culture transformation determines sustainable value creation. The convergence of edge computing, advanced caching strategies, and timeline-consistent storage technologies promises continued innovation in data architecture, enabling organizations to derive actionable insights from ever-increasing data volumes while maintaining the agility to adapt to changing business requirements. Future implementations will likely emphasize further integration between edge and cloud processing, enhanced privacy-preserving analytics capabilities, and continued democratization of data access through intuitive self-service platforms that empower users while maintaining appropriate security and governance controls.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers.

References

- [1] AltexSoft Editorial Team, (2023) Data Lake Explained: A Comprehensive Guide to Its Architecture and Use Cases, 2023. [Online]. Available: <https://www.altexsoft.com/blog/data-lake-architecture/>
- [2] DataBricks, (n.d) Delivering integrity and efficiency for the U.S. Postal Service. <https://www.databricks.com/customers/usps-oig>
- [2] Apache H B, (2021) Apache Hudi - The Data Lake Platform, 2021. [Online]. Available: <https://hudi.apache.org/blog/2021/07/21/streaming-data-lake-platform/>
- [3] DataBricks, (n.d) Driving into the future of electric transportation. <https://www.databricks.com/customers/rivian>
- [4] DataBricks, (n.d) Personalizing pharmacy care to improve patient outcomes. <https://www.databricks.com/customers/walgreens>
- [5] Fabian H and Vasiliki K, (2019) Stream Processing with Apache Flink, O'Reilly Media, 2019. [Online]. Available: <https://www.oreilly.com/library/view/stream-processing-with/9781491974285/>
- [6] GeeksforGeeks, (2024) Real-Time Analytics in Big Data, 2024. [Online]. Available: <https://www.geeksforgeeks.org/real-time-analytics-in-big-data/>
- [7] James W and Nathan M, (2015) Big Data: Principles and best practices of scalable realtime data systems, IEEE Xplore Digital Library, 2015. [Online]. Available: <https://ieeexplore.ieee.org/book/10279852>
- [8] Martin K, (2017) Designing Data-Intensive Applications: THE BIG IDEAS BEHIND RELIABLE, SCALABLE, AND MAINTAINABLE SYSTEMS, O'Reilly, 2017. [Online]. Available: [https://unidel.edu.ng/focelibrary/books/Designing%20Data-Intensive%20Applications%20The%20Big%20Ideas%20Behind%20Reliable,%20Scalable,%20and%20Maintainable%20Systems%20by%20Martin%20Kleppmann%20\(z-lib.org\).pdf](https://unidel.edu.ng/focelibrary/books/Designing%20Data-Intensive%20Applications%20The%20Big%20Ideas%20Behind%20Reliable,%20Scalable,%20and%20Maintainable%20Systems%20by%20Martin%20Kleppmann%20(z-lib.org).pdf)
- [9] Shivaram PR, (2024) Big Data Analytics: Techniques, Benefits, and Best Practices for Reliable Data, Acceldata, 2024. [Online]. Available: <https://www.acceldata.io/blog/big-data-analytics-techniques-benefits-and-best-practices-for-reliable-data>
- [10] Suma G.,(2024) What Is a Data Lake? Key Features, Use Cases, and Best Practices Explained, Acceldata, 2024. [Online]. Available: <https://www.acceldata.io/blog/what-is-data-lake-key-features-use-cases-and-best-practices-explained>